RNR-88-003

Performance Comparison of the CRAY X-MP/24 with SSD and the CRAY-2.

Richard E. Anderson
Roger G. Grimes
Engineering Scientific Services Division
Boeing Computer Services, M/S 7L-21
Seattle, WA 98124
and

Horst D. Simon[1]
Numerical Aerodynamic Simulation (NAS) Systems Division
NASA Ames Research Center, Mail Stop 258-5
Moffett Field, CA 94035

Abstract

The CRAY-2 is considered to be one of the most powerful supercomputers. Its state-of-the-art technology features a faster clock and more memory than any other supercomputer available today. In this report the single processor performance of the CRAY-2 is compared with the older, more mature CRAY X-MP. Benchmark results are included for both the slow and the fast memory DRAM MOS CRAY-2. Our comparison is based on a kernel benchmark set aimed at evaluating the performance of these two machines on some standard tasks in scientific computing. Particular emphasis is placed on evaluating the impact of the availability of large real memory on the CRAY-2 versus fast secondary memory on the CRAY X-MP with SSD. Our benchmark includes large linear equation solvers and FFT routines, which test the capabilities of the different approaches to providing large memory.

We find that in spite of its higher processor speed the CRAY-2 does not perform as well as the CRAY X-MP on the Fortran kernel benchmark. We also find that for large-scale applications, which have regular and predictable memory access patterns, a high-speed secondary memory device such as the SSD can provide performance equal to the large real memory of the CRAY-2.

[1]The author is an employee of the ESS Division of Boeing Computer Services.

Table 1: Machine Comparison (single processor)

	X-MP/24	CRAY 2
Clock (nsec)	9.5	4.1
Number of vector units	2	2
Chaining	yes	no
Peak performance (MFLOPS)	210	487
Registers	8 × 64 words	8 × 64 words
Local memory	none	16K
Number of paths to memory	3	1
Memory size (Mwords)	4	256
SSD size (Mwords)	128	none
Memory latency (cycles)	14	45(57)
Pseudo bank cycle time	-	25(41)

1 Introduction

The CRAY-2 with its 4.1 nanosecond (nsec) clock is potentially over twice as fast as the CRAY X-MP/24 [Chen 1984,Cray 1985,Neves 1987]. In addition to its superior clock speed, the CRAY-2 has a tremendous advantage in word addressable memory. The CRAY X-MP, however, is a proven machine with a mature compiler and a large set of applications programs developed especially for its architecture. Some of the key architectural features of the two machines involved are given in Table 1. Note that we have given the features of the particular machines involved in this benchmark, for example the newer X- MPs have the faster clock rate of 8.5 nsec compared to the 9.5 nsec listed in Table 1.

Recently an upgraded version of the CRAY-2 with faster memory has been introduced [Cray 1987]. The essential difference of the newer CRAY-2 in comparison to the older CRAY-2 is its faster DRAM MOS memory, which reduces the memory latency from 57 to 45 cycles. Also all DRAM CRAY-2 systems feature pseudobanking, which allows faster memory accesses and improves the performance. Each of the 128 banks of the CRAY-2 is divided in half. If there are two access in different halfs of the same bank at the same time, then the second one can proceed after 25 cycles (41 cycles on the slower machine). Pseudo banking effectively turns the 128 banks of the

CRAY-2 into 256 banks, and thus reduces the average memory latency. The CRAY-2S system, which features even faster static random access memory (SRAM) is not considered here.

Also all data in Table 1 refer to a single CPU, since we are not concerned with multitasking performance in this benchmark. We will point out how some of these features affect the relative performance of the two machines. The first set of CRAY-2 timings was obtained in March of 1987 on the CRAY-2 with serial number 2002, which is installed with the NAS project at NASA Ames Research Center in Moffett Field, California. A second set of CRAY-2 timings was obtained in February of 1988 on the same machine (serial number 2002) in order to measure improvements in the Fortran compilers. Finally a third benchmark was carried out on the new CRAY-2 (serial number 2013) at the NAS project in order to measure the effects of the upgraded memory on the machine performance. For brevity we will refer to the serial numbers of the machines involved, when discussing the older, slower memory CRAY-2 (2002) versus the newer, faster memory CRAY-2 (2013).

The X-MP/24 timings were obtained using the Boeing Computer Services' machine in Bellevue, Washington. The Boeing CRAY X-MP, one of the older X-MPs, has a clock rate slightly slower than the current rate of 8.5 nsec on the newer models.

Twenty-four FORTRAN routines were benchmarked on both machines . These computational kernels are typical of those found in scientific programming. They were assembled based on the experience at Boeing Computer Services. Assembly coded efficient implementations of these kernels for the CRAY X-MP are available in VectorPak [Boeing 1987]. The benchmark also includes large problems which are out-of-core problems on most other machines but the CRAY-2. The solution to these problems is computed in-core on the CRAY-2 and out-of-core using the SSD (solid-state-storage device) on the X-MP.

2 Benchmarking Approach

There are 24 computational kernels in the benchmark. They are listed below with a short description of what computation they perform and the reason why they were included in the benchmark.

HC1DFT performs a single one-dimensional FFT. This is an important computational kernel which usually runs into memory contention problems.

HCFFTS performs several one-dimensional FFTs simultaneously. This is an important computational kernel which is heavily used in multidimensional FFT work. This program vectorizes across the number of FFTs and avoids the memory contention problems. This computational kernel is heavily used by several groups in the Boeing company and is designed to provide high performance on a vector computer.

HC2XFT performs a two-dimensional FFT using external storage. This code uses HCFFTS for most of the computations but also performs standard FORTRAN direct access fixed length record I/O. Although I/O is usually not a consideration with the CRAY-2's large memory, HC2XFT along with HSGEXL will measure the performance balance between I/O and CPU speeds.

HSGELE solves general systems of linear equations $Ax = b$, using the best algorithm (based on matrix multiplication) for vector computers. This again is an important computational kernel which is designed to provide high performance on a vector computer. This code was modified and used in the large problem benchmark.

HSGTLE solves a single tridiagonal system of equations $Tx = b$. This code is based on an extension of the cyclic reduction algorithm and provides very efficient performance on a vector computer.

HSMMPG computes matrix-matrix products of the form $C = AB$ and $C = C \pm AB$.

HSMVPG computes matrix-vector products of the form $y = Ax$ and $y = y \pm Ax$.

HSSGTL solves several tridiagonal systems of equations $Tx = b$ simultaneously. This code vectorizes across the systems to provide high performance for such application areas as line iterative methods.

ISAMAX finds the element with the largest magnitude in a vector. This is inherently a scalar operation but some vector architectures, such as the CRAY X-MP, support this operation in vector mode.

ISCTEQ counts the number of elements in a vector that are equal to a given scalar. This is implemented in a loop with an IF THEN -END IF construct and tests the ability of the compiler to vectorize this construct. For example, the CFT 1.13 compiler does not vectorize this loop, whereas CFT 1.14 does.

SASUM sums the absolute value of the elements in a vector. This operation tests the ability to vectorize a recursive operation by separating the operations and collapsing the partial sums at the end.

SAXPY performs $y = ax + y$, where x and y are vectors and a is a scalar. This loop tests the computer's balance between memory references (two fetches and one store) and floating point operations (one multiply and one add). The X-MP can execute two fetches and a store concurrently, while the CRAY-2 can execute only one memory access instruction at a time (either a fetch or a store).

SAXPYI performs an indexed SAXPY. This kernel performs the above loop, except an index array is used for referencing the elements of the vector x. This kernel tests the ability of the machine to randomly fetch entries from memory.

SCOPY copies vector x into vector y. This tests speeds of memory references. The X-MP can do a fetch and a store concurrently while the CRAY-2 must execute the fetch and store separately.

SDOT computes the vector inner product. This is an important computational kernel which requires both a balance between memory and floating point as well as the ability to collapse partial sums.

SDOTI performs an indexed SDOT.

SGTHR gathers entries of vector x specified by an index vector into the dense vector y.

SLSTNE counts and lists entries of a vector not equal to a scalar. This is similar but more complex than ISCTEQ.

SNRM2 computes the Euclidean norm.

SSCAL scales a vector with a scalar.

SSCTR scatters the entries of a dense vector into specified entries of a vector x. This is the reverse of SGTHR.

SSWAP interchanges the contents of two vectors. This is a memory intensive operation requiring two fetches and two stores with no floating operations involved. This tests the memory reference speeds of the computer.

All the subroutines were written in portable FORTRAN 77. They were compiled with CFT 1.13 on the X-MP. The exceptions were SGTHR and SSCTR, which require CFT 1.14 to vectorize gather/scatter instructions. The code was written as portable FORTRAN and no attempt was made to take advantage of the X-MP's architecture or compiler. (For example, there was no unrolling of outer DO LOOPs.) These kernels are designed to test the performance of a given computer/compiler in executing FORTRAN.

The same Fortran routines were ported to the CRAY-2 in March 1987 and compiled with CFT77, which is a port from CFT 1.09. Neither the X-MP nor the CRAY-2 would vectorize the complex SAXPY, so a compiler directive (IVDEP) was added and both rates are reported. The CRAY-2 did vectorize the complex dot product while the X-MP would not. A compiler directive was added to the complex dot product and both times are reported for the X-MP.

In March 1987 code compiled under CFT2 generally executed about 30% faster than the same code compiled with CFT77. Unfortunately we were unable to use CFT2 because at the time of the original benchmarking CFT2 did not support generic functions (such as MIN, MAX, and LOG). The July 1987 release of CFT2 (version 3.0b) does support generics. However, the most recent release of CFT77 (version 1.3) is now within 5% of the performance of CFT2. Therefore there we did not include benchmarking results obtained with CFT2.

Table 2: **Performance Comparison of Two Compilers on the CRAY-2 Using a Two-dimensional FFT Routine.**

Problem Size	CFT77 MFLOPS	CFT2 MFLOPS	Ratio CFT2/CFT77
1024 × 1024	91	120	1.3
1024 × 2048	95	130	1.4

The potential performance variation due to different compilers can be seen in the following example. Bailey's FORTRAN FFT code was used to compare the two compilers CFT77 (version 1.2) and CFT2 in March 1987 [Bailey 1987]. The results are given in Table 2. The 35% performance difference between CFT77 and CFT2 was typical of results obtained using the two different compilers at that time.

3 Performance on the Kernel Benchmark

Table 3 shows the performance for the routines described in section 2. The X-MP timings were obtained at three different times under various system loads and the variation in timings was less than 10%. The CRAY-2 benchmark was executed ten times at various system loads and most routines showed a variation over 30% from best to worst. The times reported for the CRAY-2 are the median from all the data collected. The CRAY-2 data were obtained on serial number 2002 with CFT77, version 1.2.

The kernels which operated on vectors were tested with vector lengths from 1 to 20,480. The rates reported are for the vector lengths of 20,480. The matrix-vector and matrix-matrix operations used vectors with lengths ranging from 1 to 256. These rates were computed with the longest vector as well.

The table also provides the vector half-performance length, i.e., the length at which the vector achieves one half of the maximum performance for that operation. The true asymptotic rate was not actually determined, but taken from the vector performance at 20,480. The CRAY-2 generally reaches its half-performance length before the X-MP, but this can be attributed to the X-MP achieving higher rates.

On the CRAY-2, a timing problem exists for the gather instruction, so the hardware forces a maximum memory reference which results in three null references for each data reference. As a result, SGTHR timings are much slower than SSCTR (scatter) timings.

Note that in Table 3 SGTHR and SSCTR have been compiled using CFT 1.14 on the CRAY X-MP in order to take advantage of the hardware gather/scatter. Also the "asymptotic" rate for the matrix operations is the performance for $n = 256$.

Simply considering the basic clock speed one would expect the CRAY-2 to be about twice as fast as the X-MP. The most important result of this benchmark, which becomes clear from Table 3, is that this is *not* the case. In many examples the CRAY-2 performed more slowly than the CRAY X-MP, in some extreme cases by as much as a factor of three. Most of the comparatively slow rates can be explained by considering the impact of architectural characteristics of the CRAY-2.

There were three routines which executed without vectorization on both machines: ISAMAX, ISCTEQ, and SLSTNE. Here the faster speed of the CRAY-2 was significant and it outperformed the X-MP on these problems. These were the only kernels that consistently ran faster on the CRAY-2 than the X-MP.

In executing the routines that vectorized, the CRAY-2 had slight advantage in a few instances (e.g. SSWAP), but most routines were significantly faster on the X-MP. For example, on HSSGTL (tridiagonal solver), the X-MP ran more than three times faster than the CRAY-2.

On one group of routines the speed of the X-MP can be attributed to the availability of its three paths to memory; only one path is available on the the CRAY-2. The SAXPY kernel could potentially be performed three times as fast on the CRAY-2 with three paths and chaining available. This would bring the SAXPY performance on the CRAY-2 in about the right range relative to its clock speed. Since SAXPY-type operations are predominantly used in HSGELE, HSMMPG, and HSMVPG these kernels could also perform significantly faster with three paths.

For some other kernels the main source of performance degradation on the CRAY-2 were due to memory bank conflicts. This applied to the FFT routines (HCFFTS, HC1DFT, and HC2XFT). Table 3 also lists many rou-

Table 3: **Results of the Kernel Benchmark.**

Subroutine (stride)	CRAY-2 Rate	$N_{\frac{1}{2}}$	CRAY X-MP Rate	$N_{\frac{1}{2}}$	X-MP/CRAY-2 Ratio
HCFFTS	45.6		80.6		1.77
HC1DFT	4.8		13.7		2.85
HC2XFT	37.7		69.9		1.97
HSGELE	38.0		74.8		1.97
HSGTLE	28.3		37.0		1.31
HSMMPG	48.3	46	92.7	76	1.92
HSMVPG	50.3	39	99.7	61	2.00
HSSGTL	3.2		10.5		3.28
ISAMAX (1)	2.0	6	1.5	5	.72
ISAMAX (32)	1.9	6	1.2	5	.62
ISCTEQ (1)	2.7	6	1.6	3	.62
ISCTEQ (32)	2.2	6	1.5	2	.71
SASUM (1)	53.0	228	78.9	459	1.49
SASUM (32)	16.5	86	24.6	152	1.50
SAXPY (1)	59.0	89	131.5	225	2.20
SAXPY (32)	10.0	16	17.4	34	1.74
SAXPYI	2.2	4	4.8	6	2.18
SAXPYI (CDIR)	22.0	32	43.0	56	1.95
SCOPY (1)	85.0	94	148.0	223	1.74
SCOPY (32)	14.0	43	12.4	41	.89
SDOT (1)	77.0	219	140.0	451	1.83
SDOT (32)	18.8	56	24.2	85	1.29
SDOTI	27.0	79	7.7	18	.28
SDOTI (CDIR)	27.0	79	60.0	160	2.22
SGTHR	51.0	31	106.0	89	2.08
SLSTNE (1)	2.1	6	1.5	3	.70
SLSTNE (32)	2.1	6	1.4	3	.70
SSCAL (1)	41.0	80	68.3	157	1.67
SSCAL (32)	8.0	16	12.2	27	1.53
SSCTR	90.0	40	112.0	92	1.24
SSWAP (1)	54.0	61	50.2	87	.93
SSWAP (32)	8.0	10	12.1	25	1.51

tines with stride 1 and stride 32. The stride 32 is the worst case stride for the X-MP. The worst case for the CRAY-2 is 256 (considering pseudobanking); however, a 32 stride causes a bank conflict every fourth clock cycle on the CRAY-2. The bank resolution time on the CRAY-2 is 45(or 57 for the slower memory machine) clock cycles and 4 clock cycles on the X-MP, so the penalty for bank conflicts is much more severe on the CRAY-2. Both computers showed substantial degradation with the stride 32. However, the degradation was comparatively greater on the CRAY-2 as, for example, in SSWAP.

The worst performance ratio (3.28 times slower) for the CRAY-2 was obtained for the tridiagonal linear equation solver. However, this Fortran code is based on a cyclic reduction algorithm, involving parameters which have been optimized for the CRAY X-MP.

The performance of the CRAY-2 has been significantly improved over the last year. Major improvements are the initially mentioned upgrade to faster memory, as well as new releases of the Fortran compilers. We repeated the above benchmark in February 1988, and took both new improvements into account. The results are given in Table 4. The combined effect of both a better compiler and a faster memory resulted in up to 40% performance improvement. In some cases this was enough for the CRAY-2 to come close or even surpass the CRAY X-MP performance.

4 Performance on Large Memory Applications

Two standard linear algebra subroutines were chosen to evaluate the performance of both machines when large memory is required. In particular we were interested in benchmarking a code that required SSD usage on the X-MP. Contrary to the previous section in which Fortran kernels were compared on the two machines, here we attempted to use code which has been optimized for each machine. All CRAY-2 results in this section were obtained on the slower memory machine serial number 2002, using CFT version 1.2.

As the first benchmarking task a two-dimensional complex FFT was chosen for the following reasons: it is an important kernel in applications programming, it requires a large amount of both computation and I/O, and it

Table 4: **Performance Improvements on the CRAY-2 through Compiler and Hardware Upgrades (all figures are MFLOPS).**

Subroutine	Serial 2002 CFT77 version 1.2	Serial 2002 CFT77 version 1.3	Serial 2014 CFT77 version 1.3	X-MP CFT1.13
HCFFTS	45.6	53.0	58.0	80.6
HC1DFT	4.8	5.8	8.0	13.7
HC2XFT	37.7			69.9
HSGELE	38.0	44.0	48.0	74.8
HSGTLE	28.3	32.0	40.0	37.0
HSMMPG	48.3	59.0	60.0	92.7
HSMVPG	50.3	60.0	60.0	99.7
HSSGTL	3.2	4.8	4.8	10.5
ISAMAX (1)	2.0	2.0	2.0	1.5
ISAMAX (32)	1.9	1.9	2.0	1.2
ISCTEQ (1)	2.7	2.3	2.5	1.6
ISCTEQ (32)	2.2	2.3	2.4	1.5
SASUM (1)	53.0	72.0	78.0	78.9
SASUM (32)	16.5	16.0	22.0	24.6
SAXPY (1)	59.0	71.0	82.0	131.5
SAXPY (32)	10.0	11.0	16.0	17.4
SAXPYI	2.2	4.5	5.0	4.8
SAXPYI (CDIR)	22.0	26.0	29.0	43.0
SCOPY (1)	85.0	63.0	75.0	148.0
SCOPY (32)	14.0	7.6	15.0	12.4
SDOT (1)	77.0	85 .0	98.0	140.0
SDOT (32)	18.8	18.0	25.0	24.2
SDOTI	27.0	34.0	39.0	7.7
SDOTI (CDIR)	27.0			60.0
SGTHR	51.0	72.0	83.0	106.0
SLSTNE (1)	2.1	2.3	2.3	1.5
SLSTNE (32)	2.1	2.0	2.5	1.4
SSCAL (1)	41.0	46.0	46.0	68.3
SSCAL (32)	8.0	8.5	12.0	12.2
SSCTR	90.0	77.0	88.0	112.0
SSWAP (1)	54.0	58.0	72.0	50.2
SSWAP (32)	8.0	7.3	14.0	12.1

Table 5: **Performance Comparison on Two-dimensional FFT Codes.**

Dimension	CRAY-2 MFLOPS	X-MP MFLOPS	CRAY-2/X-MP
1024 × 1024	123	95	1.3
1024 × 2048	130	95	1.4
1024 × 4096	136	97	1.4
2048 × 2048	142	96	1.5
2048 × 4096	146	96	1.5
4096 × 4096	148	95	1.6

vectorizes well by performing simultaneous one dimensional FFTs.

The two-dimensional FFT executed on the X-MP is an optimized CAL code that writes intermediate results to the SSD [Boeing 1987]. On the CRAY-2 we used an FFT developed by Bailey. Bailey's one-dimensional FORTRAN FFT is a radix-4 algorithm that avoids power of two memory strides [Bailey 1987]. Although this comparison may appear unfair, i.e., CAL on the XM-P and FORTRAN on the CRAY-2, it should be noted that Bailey's one-dimensional FFT outperforms the CAL version supplied by CRAY Research. For the two-dimensional version the CAL coded routine in CRAY's SCILIB runs at 300 MFLOPS and is faster than Bailey's two-dimensional FFT. The two-dimensional FFTs demonstrate what the CRAY-2 was designed to do, namely to provide an environment for developing portable code without the complexities of disk I/O and assembly programming. Table 5 shows the results of this benchmark comparison.

The second application kernel for benchmarking was a general linear equation solver. The general dense linear equations were solved on the CRAY-2 using a modified version of SGEFA from LINPACK [Bunch et. al. 1979]. The initial version, a straight-forward FORTRAN implementation, ran only about 36 MFLOPS on the CRAY-2. Then the FORTRAN subroutine in SGEFA performing matrix-vector multiplications was replaced with the corresponding CAL routine MXVA. A considerable increase in performance resulted as shown in Table 5. However, the times reported here for the CRAY-2 should not be taken as optimal for solving linear equations on the CRAY-2. They are just an indication of the performance one can achieve on large problems by making a few simple modifications in existing FORTRAN code. A

Table 6: **Performance Comparison on Linear Equation Solver.**

Size	Rate (worst)	CRAY-2 Rate (best)	CRAY-2 best/worst	X-MP Rate
1000	151	229	1.5	190
1500	152	246	1.6	191
2000	157	255	1.6	191
2500	153	260	1.7	192
3000	156	264	1.7	191

linear equation solver that operates at over 350 MFLOPS on large problems has been developed by Calahan [Calahan 1986]. An even faster solver might be possible by using Calahan's approach based on a matrix-matrix multiplication kernel, and by utilizing the new fast matrix-matrix multiplication subroutine developed by Bailey [Bailey 1988]. CRAY's SCILIB provides a routine for matrix inversion, which runs at 300 to 400 MFLOPS.

The times reported in Table 6 for the X-MP, however, are optimal. The linear equation solver used on the X-MP is using an out-of-core Gaussian elimination algorithm, based on block matrix-matrix products [Grimes 1987]. The program is running at about 90% of peak machine speed and implemented as HSGEXL in VectorPak [Boeing 1987].

The linear equation solver was executed five times on both machines. The remarkable result in Table 6 is not so much the actual performance, but the considerable performance variation on the CRAY-2. While all the routines varied about 15 to 35% in performance depending on system load, a 70% difference was noted in the linear equation solver. The best case times were obtained on a Sunday morning at 2:44 A.M. The machine was probably idle at that time so memory contentions were minimal. The worst case times reported are closer to the average and to what one would expect to get when the machine is busy. (All the times reported here are averages of five runs. Each problem size is run five times consecutively.)

5 Conclusions

On the Fortran kernel benchmark the performance of the CRAY-2 varied from about a third the performance of the CRAY X-MP to three times the performance of the X-MP. In many instances the comparatively worse performance of the CRAY-2 can be directly attributed to architectural disadvantages most notably the limited one path to memory, and the relatively slow memory. Improvements in compilers and improvements in memory speed have led to some considerable overall performance improvements on the CRAY-2. However, both will not be able to overcome some of the architectural limitations of the CRAY-2.

While the X-MP generally had an advantage on the Fortran kernels, the CRAY-2 showed it could easily outperform the X-MP on large problems. The CRAY-2 did this without the extra effort of writing temporary results to a disk file. However, the better performance did not come as easily as a general user would hope for. Fast algorithms for the CRAY-2 require a detailed understanding of the architecture of the machine, and a fair amount of sophistication when implemented (see [Bailey 1987,Bailey 1988,Calahan 1986]). We venture to say here that the programming effort in implementing high speed linear algebra algorithms for the CRAY-2 can be of the same level of difficulty, as the corresponding effort in implementing out-of-core algorithms using the SSD on the CRAY X-MP. For computations which require a regular and predictable access to the data, the X-MP type architecture with a high-speed secondary memory device (which has evolved by now to the new CRAY Y-MP) is an efficient alternative to the large real memory of the CRAY-2. But obviously the CRAY-2 is the machine of choice for any more complicated application program with large real memory requirements, for which a rewriting using I/O operations is out of question.

References

[Bailey 1987] D. H. Bailey. *A High Performance FFT Algorithm for Vector Supercomputers.* Technical Report, NAS Systems Division, NASA Ames Research Center, Moffet Field, California, July 1987. submitted to Journal of

Supercomputer Applications.

[Bailey 1988] D. H. Bailey. Extra high speed matrix multiplication on the CRAY-2. *SIAM J. Sci. Stat. Comp. (to appear)*, 1988.

[Boeing 1987] Boeing Computer Services. *VectorPak Users Manual.* 1987. Document No. 20460-0501-R1.

[Bunch et. al. 1979] J. Bunch, J. Dongarra, C. Moler, and G. Stewart. *LINPACK User's Guide.* SIAM Publications, Philadelphia, 1979.

[Calahan 1986] D. A. Calahan. Block oriented, local-memory-based linear equation solution on the CRAY-2: uniprocessor algorithms. In *Proceedings of the 1986 International Conference on Parallel Processing*, pages 375 – 378, IEEE Computer Society, Los Angeles, 1986.

[Chen 1984] Steve S. Chen. Large-scale and high-speed multiprocessor system for scientific applications:Cray X-MP series. In J. Kowalik, editor, *High-Speed Computation*, pages 59 – 68, Springer Verlag, NATO ASI Series, Heidelberg, 1984.

[Cray 1985] Cray Research, Inc. Introducing the CRAY-2 computer system. *CRAY Channels*, 2 – 5, Summer 1985.

[Cray 1987] Cray Research, Inc. Introducing the enhanced CRAY-2 series of computer systems. *CRAY Channels*, 2 – 3, Summer 1987.

[Grimes 1987] R. Grimes. *Solving Systems of Large Dense Linear Equations.* Technical Report ETA-TR-44, Boeing Computer Services, Seattle, Washington, 1987. to appear in The Journal of Supercomputing.

[Neves 1987] Kenneth W. Neves. *Supercomputing: Hardware and Algorithms.* Technical Report ETA-TR-62, Boeing Computer Services, Seattle, Washington, 1987.

Richard E. Anderson is a member of the Applied Mathematics Group at Boeing Computer Services. Before joining Boeing he worked in instrumentation and control at the Stanford Linear Accelerator Center. His work involves optimizing existing numerical software for supercomputer architectures as well as benchmarking and code development.

Anderson earned a B.S in Mathematics from Brigham Young University and a M.S. in Mathematics from Montana State University.

Roger G. Grimes was born in Omak, Washington, on August 5, 1953. He received his M.A. degree in Mathematics from The University of Texas at Austin, Texas, in 1976.

In 1977 he joined the Center for Numerical Analysis at The University of Texas as a mathematical software programmer. His main area of work was iterative methods for the solution of real symmetric positive definite linear systems. In 1979 he joined Boeing Computer Services as a numerical analyst. Since then he has been involved with the development and the application of mathematical software for a wide variety of areas including the solution of large linear algebra problems, both dense and sparse.

Horst D. Simon was born in Stadtsteinach, West Germany, on August 8, 1953. He received his Diplom in mathematics from the TU Berlin in 1978, and his Ph.D. in Mathematics from the University of California, Berkeley, in 1982.

After a year as assistant professor for applied mathematics at SUNY Stony Brook, he joined Boeing Computer Services in 1983 as a research analyst. His assignments included the development of mathematical software for the CRAY 1-S and CRAY X-MP computers, and the coordination of research and consulting efforts for Boeing's participation as a Phase I site in the NSF supercomputing initiative. From 1986 to 1987 he was manager of the Computational Mathematics Group, which include the management of the Algorithm Research Project at Boeing's High Speed Computing Center. In Fall of 1987 he began a new assignment with the NAS project at NASA Ames Research Center, Moffett Field, California.